Lecture 2

CS 6260 Automated Planning and Learning

Deterministic Search

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Some material based on lecture slides for ASU's *CSE571 Artificial Intelligence by Siddharth Srivastava*.

Intro and Logistics (Recap)

Instructor: Pulkit Verma

- Preferred mode of contact:
  pulkitv@cse.iitm.ac.in

- PhD from **Arizona State University**
- Postdoc from **MIT CSAIL**

- Research Areas:
  Automated Planning, Human-AI Collaboration, AI Safety and Assessment, Robotics, Lifelong Learning, …

pulkitverma.net

Teaching Assistants

Sayak

Swapnil

Tushar

Lectures

- CS 15, CSB Ground Floor (of course!!)
- Tu: 11-11:50 AM; W: 10-10:50 AM; Th: 08-08:50 AM; F: 05-05:50 PM
  - Some classes will be cancelled due to official travel; watch out for announcements.
- Attendance is mandatory
- **Office Hours**: Starting Feb 1st week. Watch out for announcement.

Resources

- Course Website: hails-group.github.io/courses/cs6260/jm26.html
  - Slides posted here

- Announcements: Moodle

- Course Email: cs6260.jm26@gmail.com
  - No URGENT email responses. Please **plan** ahead.
  - We will respond to each email within 48 hours.
  - Any course-related query should be posted on Moodle so that others can also benefit from the answer.

Exams

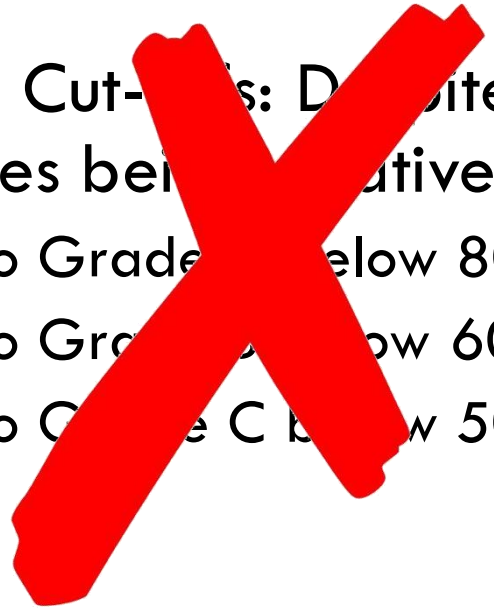
- 3 Exams:
  1. [Quiz 1] Feb 19: 08:00 AM – 08:50 AM
  2. [Quiz 2] Apr 02: 08:00 AM – 08:50 AM
  3. [End Sem] May 12: 08:00 AM – 08:50 AM

- More details closer to exams.

Grading Breakup

- Exams: 70%
  1. Quiz 1 : 20%
  2. Quiz 2 : 20%
  3. End Sem: 30%
- Assignments: 25%
- In-class Tasks: 5%

- Relative Grading
- Hard Cut-offs: Definite grades below relative:
  - No Grade below 80%
  - No Grade below 60%
  - No Grade C below 50%



Assignments

- n (n=4 or 5) Assignments. Equal distribution, i.e., each assignment will be of (25/n)%.

- Life Happens!!
  - Late submission (up to 2 days) allowed on any one. No questions asked. Just let us know!!
  - If facing any emergency, let us know; we can work something out. We are here to help.

Prerequisites

- Probability
- First Order Logic / Logic in CS
- Programming (Python)
- Analysis of Algorithms

- Sample Knowledge Check: https://forms.gle/PnMCm4qy2DuLxQaq7

- Option to drop the course if you feel like its not for you

Academic Integrity

- Ask us if you ever need anything.
  - Office Hours, in addition to regular classes, if you are stuck.
  - Post on Moodle or Email us: No question is too silly or obvious.
  - Collaboration is okay. But final answers should be yours. Cite any collaborators.
  - Do not share your answers with anyone, and do not post any solutions online.

- No relief if any violation found!!
  - Will receive 0 on the assignment/quiz/exam.
  - U grade, if needed.

- Drop the course if you feel you don't know

We are here to Help

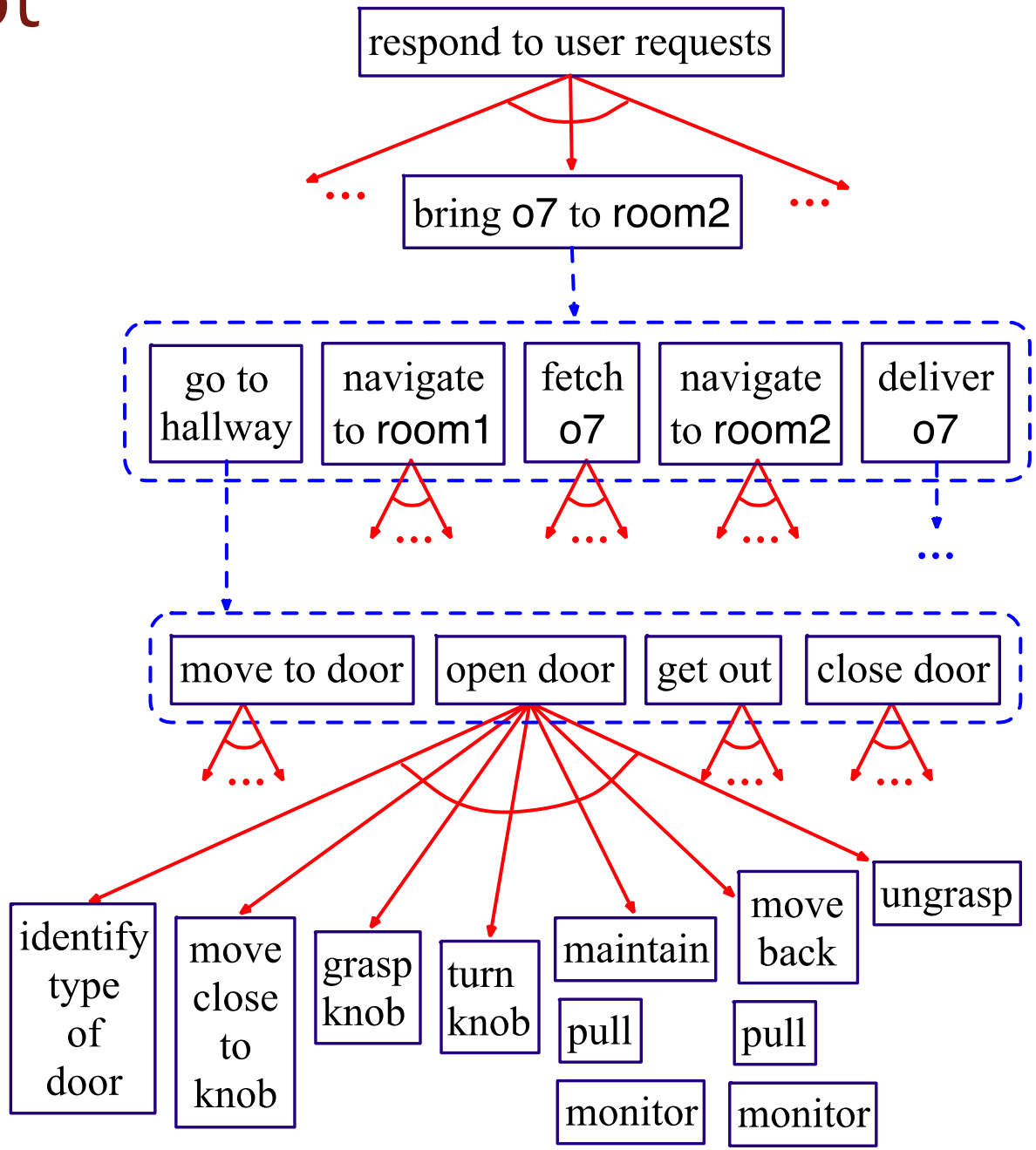
- Ask us if you ever need anything.
  - Office Hours, in addition to regular classes, if you are stuck.
  - Post on Moodle or Email us: No question is too silly or obvious.

- Wellness Center: https://dost.iitm.ac.in/iitmdost/wellness-center

- Student Wellness Resources: https://dost.iitm.ac.in/iitmdost/student-wellness-resources-overview

Recap: Service Robot

- Multiple levels of abstraction
  - Higher levels: more planning
  - Lower levels: more acting
- Continual online planning
  - What room is o7 in?
  - What route?
  - What kind of door?
  - Close enough to door handle?
- Heterogeneous reasoning
  - planning abstract tasks
  - path planning
  - reactive (e.g., open door)



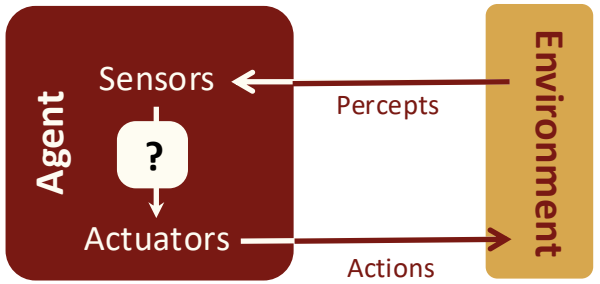
Planning

Acting

Agents

Agents

- Agent function
  - Maps percept histories to actions:
  - $f: P^* \rightarrow A$



- Agent function needs to be computed
  - Agent program: the program that computes f for a given agent and agent function

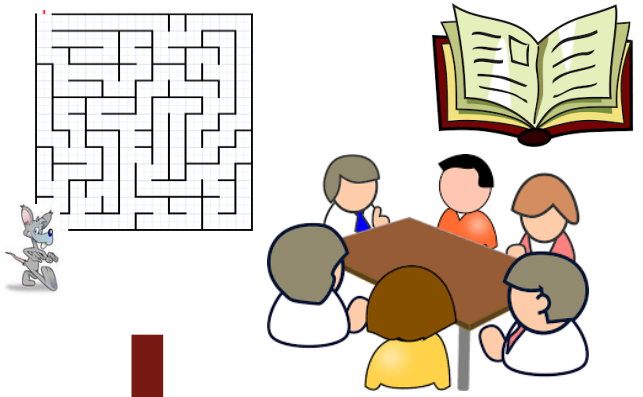
- Agent behavior is encoded by the agent function

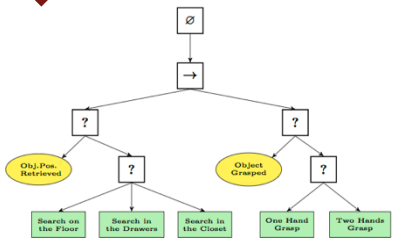
The Typical Agent Design Process

1. **Represent** background information and objective
   - Multi-step agents require states, actions and their effects on states
   - Learned using simulator, modeled using literature/experts, or partially modeled with scope for online learning

2. **Compute** "behavior" that would allow agent to achieve objective
   - Behavior may include contingencies depending on observed system state during execution

3. **Execute** this behavior
   - The bottom line, drives all of the above





Model-based Agents and Model-free Agents

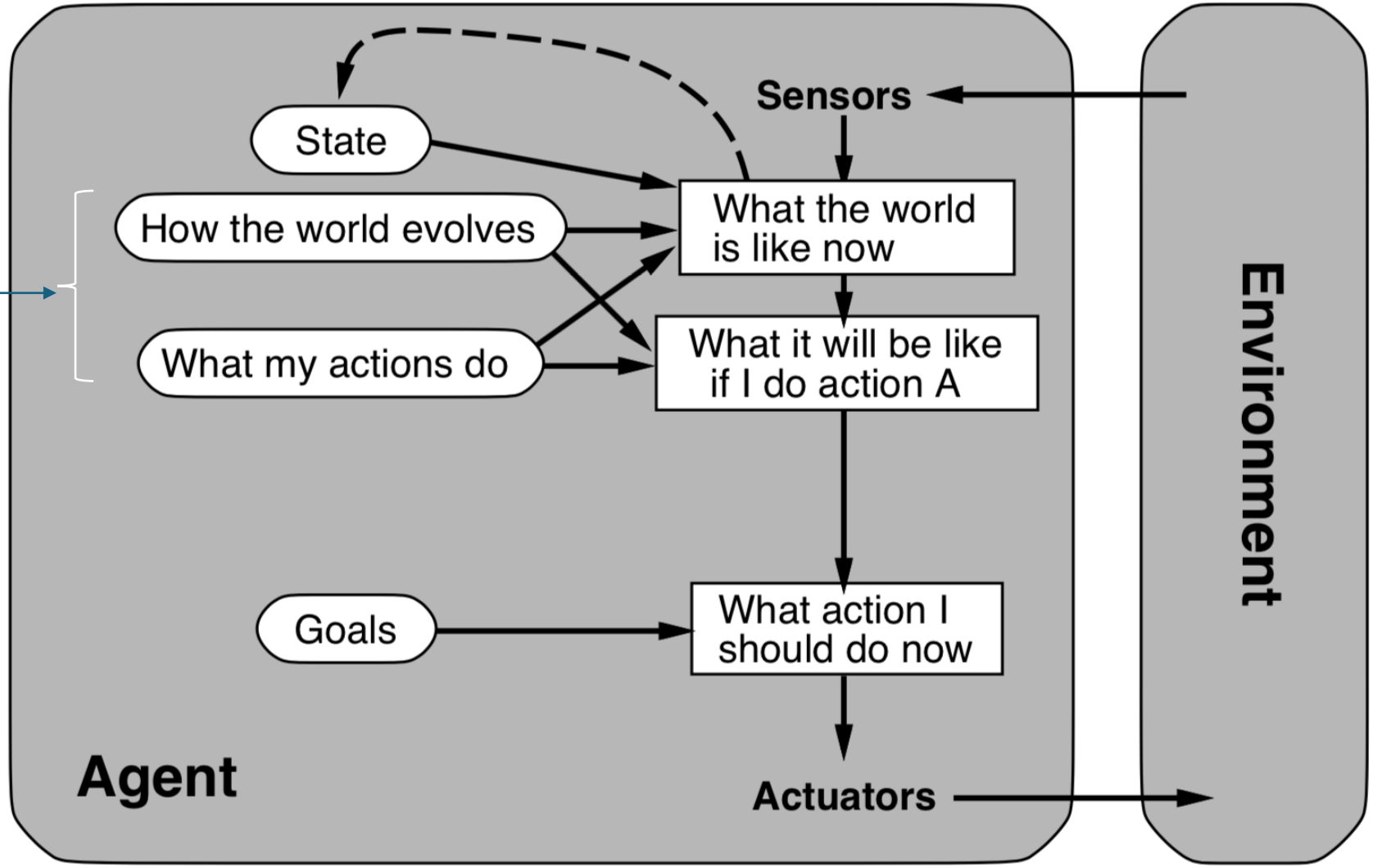
- Model-free agents evaluate courses of action directly through interaction with the real world
  - Rare, due to safety and quality concerns

- Model-based agents evaluate courses of action "in their heads"
  - May employ different types of models:
    - simulators (arbitrary generative models) that provide samples from the possible results of an action
    - analytical models that provide precise sets of possible results of actions
  - E.g., Atari-game-playing agents, AlphaGo, etc.

- Ideal situation: initialize agents using available models; improve through learning on-the-fly

Goal-based Agents

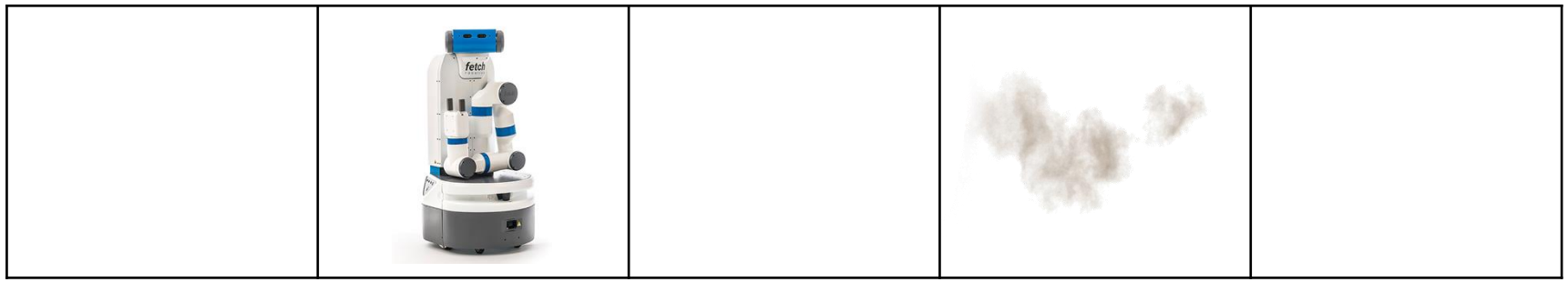
Models:
Simulators or
Analytical specs



Types of Agents and Environments

Problem Types and Solution Representations

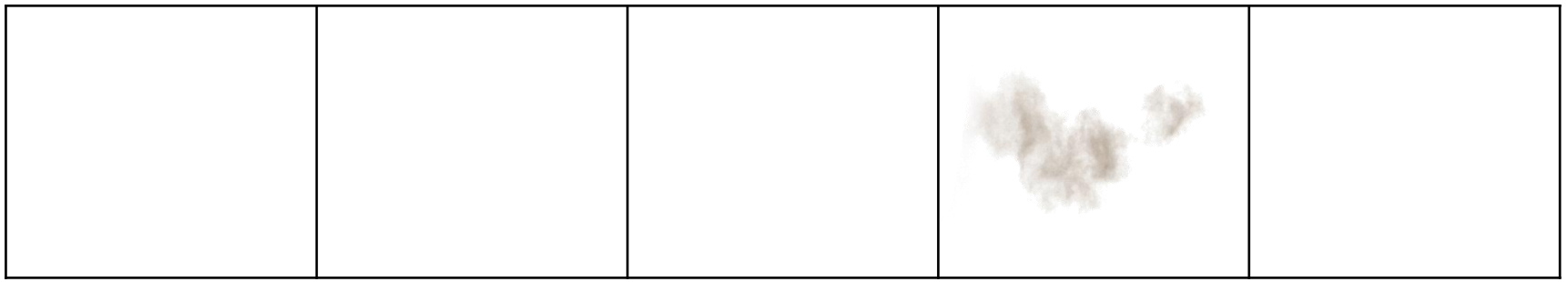
- Deterministic, fully observable problem
  - Actions are deterministic; agent observes full state
  - "Classical planning" or search: solution is a plan, or a sequence of actions



Solution plan: right, right, suck

Forms of Solutions

- Deterministic, fully observable
- Deterministic, non-observable
  - "Conformant planning": solution is a plan, but often no solutions exist



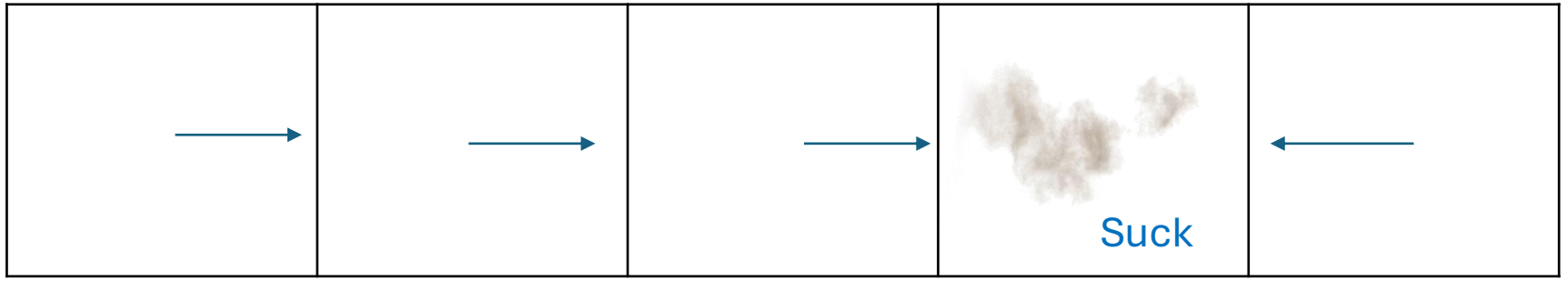
???

Right, Right, Right, Right, Suck, Left, Suck, Left, Suck, Left, Suck, Left, Suck

Forms of Solutions

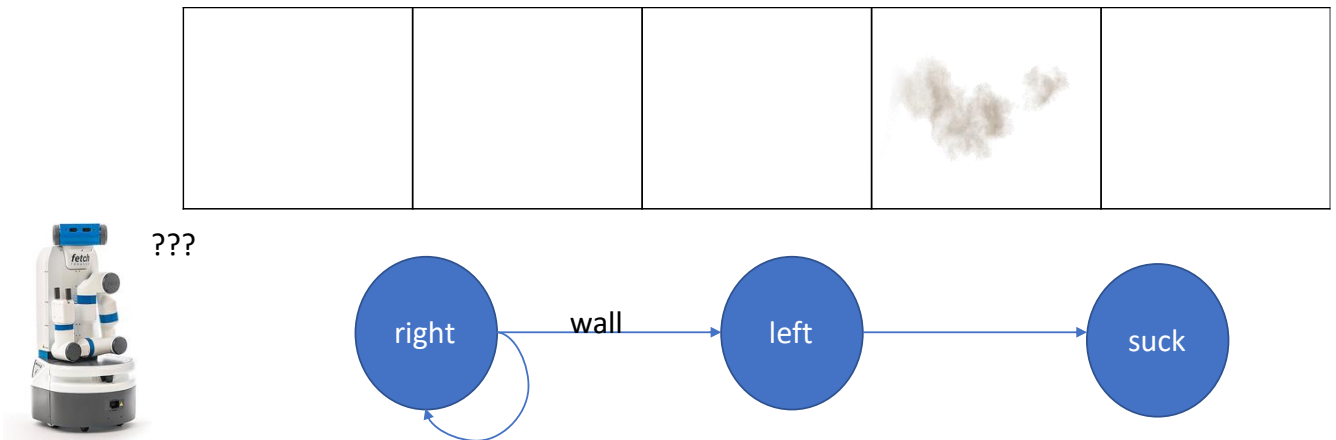
- Deterministic, fully observable
- Deterministic, non-observable
- Non-deterministic, fully observable:
  - Actions may have multiple possible outcomes (noisy actuators, exogenous events)
  - "Contingent planning": solution can be a policy (a function $S \rightarrow A$)

Non-determinism: a move action may fail



Forms of Solutions

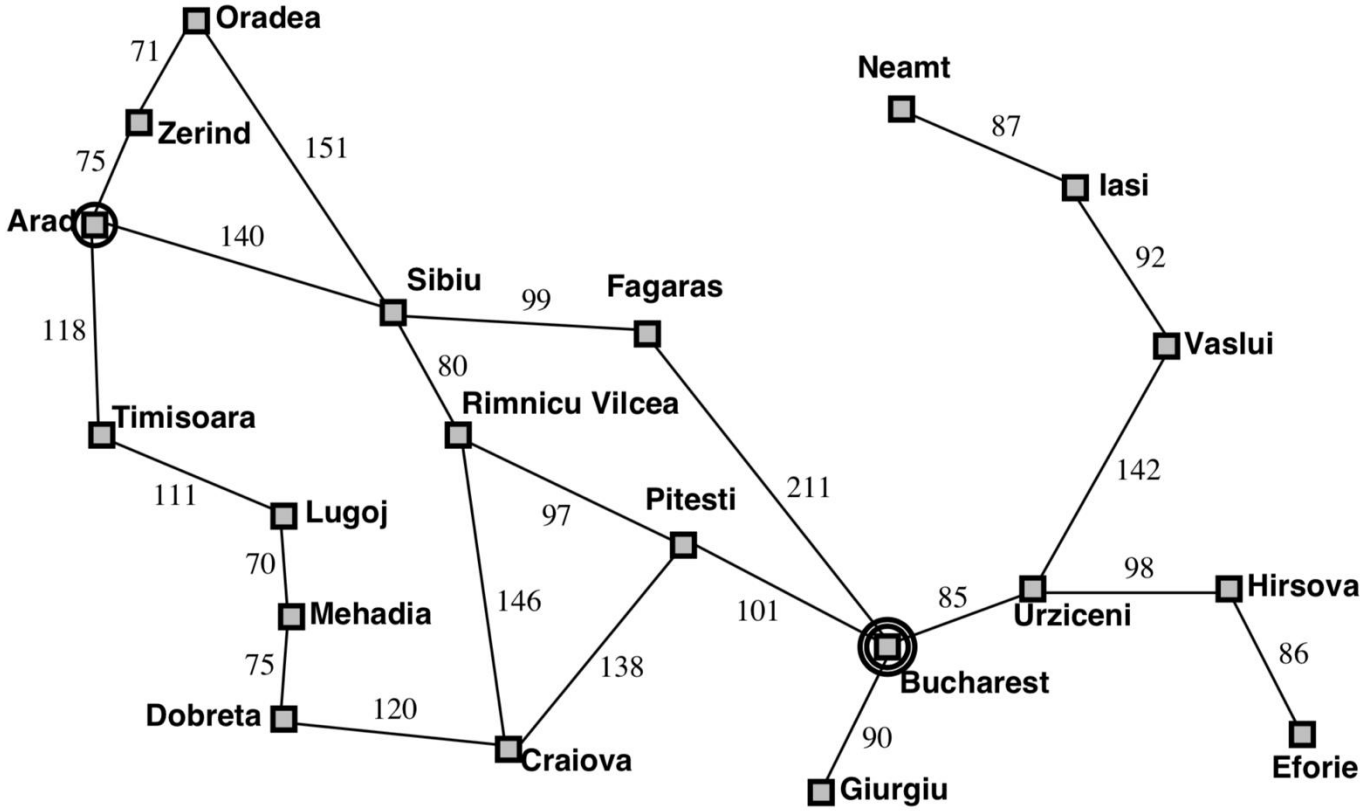
- Deterministic, fully observable
- Deterministic, non-observable
- Non-deterministic, fully observable:
- Partially observable
  - Sensors report noisy, incomplete information about the current state
  - Solution is a finite state machine



Deep Dive: Deterministic and Fully Observable Systems

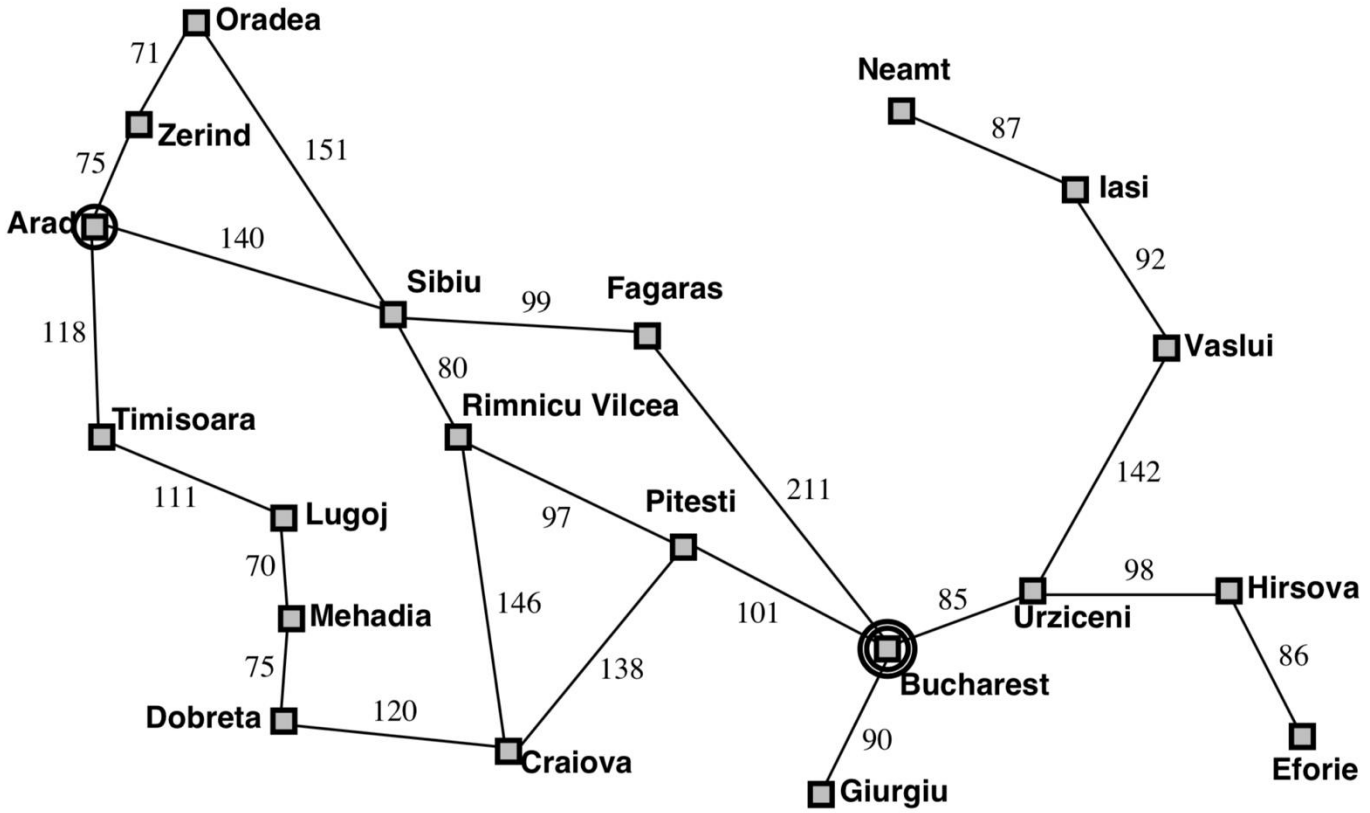
Route-Planning Agent

- State:
  - <current city>
- Action:
  - Move along edge
  - Transition function:
    - $T: S \times A \rightarrow S$
- Performance measure:
  - Cost (length) of path from initial city to goal city
- Such problems are known as *search problems*



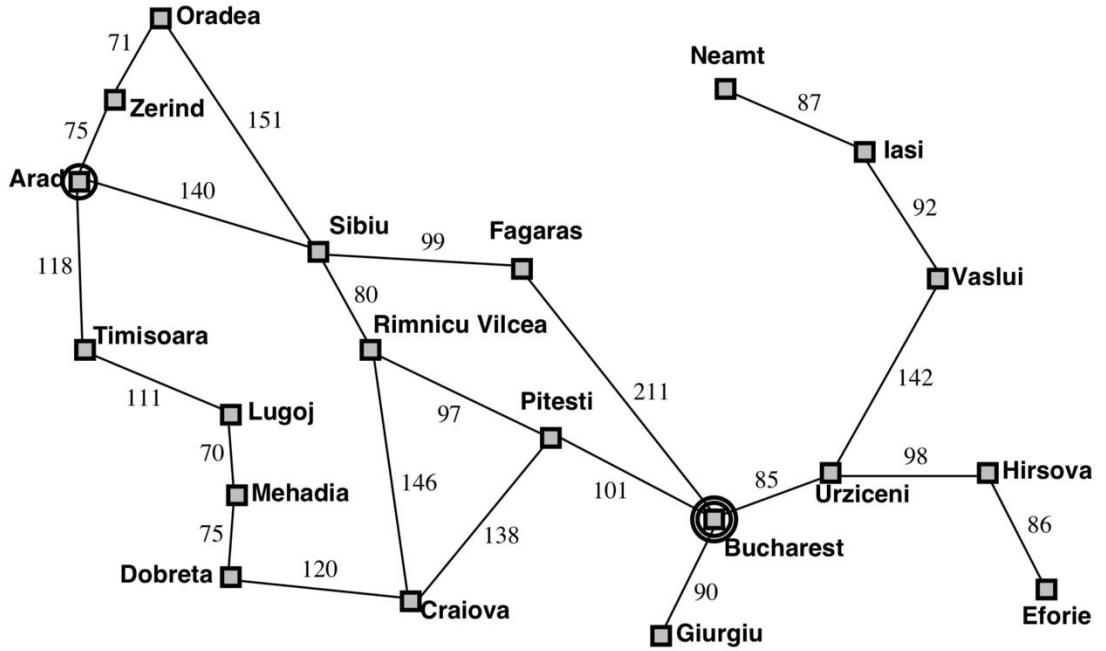
Route-Planning Agent

- Examples of search algorithms?



Scope of Search Problems

- Search problems go beyond route planning
- All that's needed:
  - Deterministic actions
  - Fully observable state
  - Initial state, goal state(s)
- What types of problems can be expressed as a graph?



General Form of Search Problems

Household Robot as an Agent

Possible Actions

Move_arm(TargetPose)

Move_base(TargetBasePose)

Run_object_detection(currentFOV)

Toggle(WiFiLightSwitch, On)

MoveCamera(P,T,Z)

...

Result States

...

x

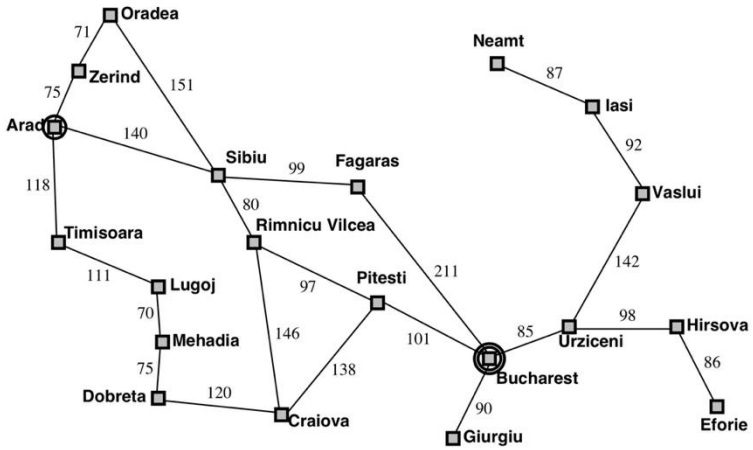
x

t=0

Decision Points

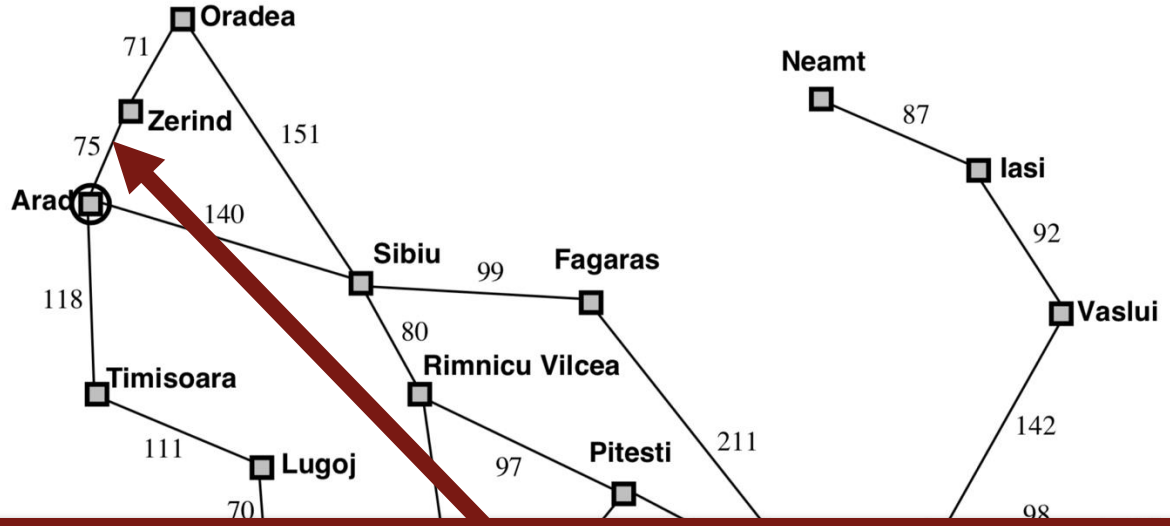
Naïve Formulation of Search Problems

- Given:
  - Graph $\langle V, E, A, \ell \rangle$
    - V set of vertices
    - E set of directed edges
    - A set of actions
    - $\ell: E \rightarrow \langle A, \mathbb{R}^+ \rangle$
  - $v_i \in V$ start vertex
  - G goal vertex set
- Find a path from v_i to G



Naïve Formulation of Search Problems

- Given:
  - Graph $\langle V, E, A, \ell \rangle$
    - V set of vertices
    - E set of directed edges
    - A set of actions
    - $\ell: E \rightarrow \langle A, \mathbb{R}^+ \rangle$
  - $v_i \in V$ start vertex
  - G goal vertex set
- Find a path from v_i
- The directed edge f



(Arad, Zerind)
(Zerind, Arad)

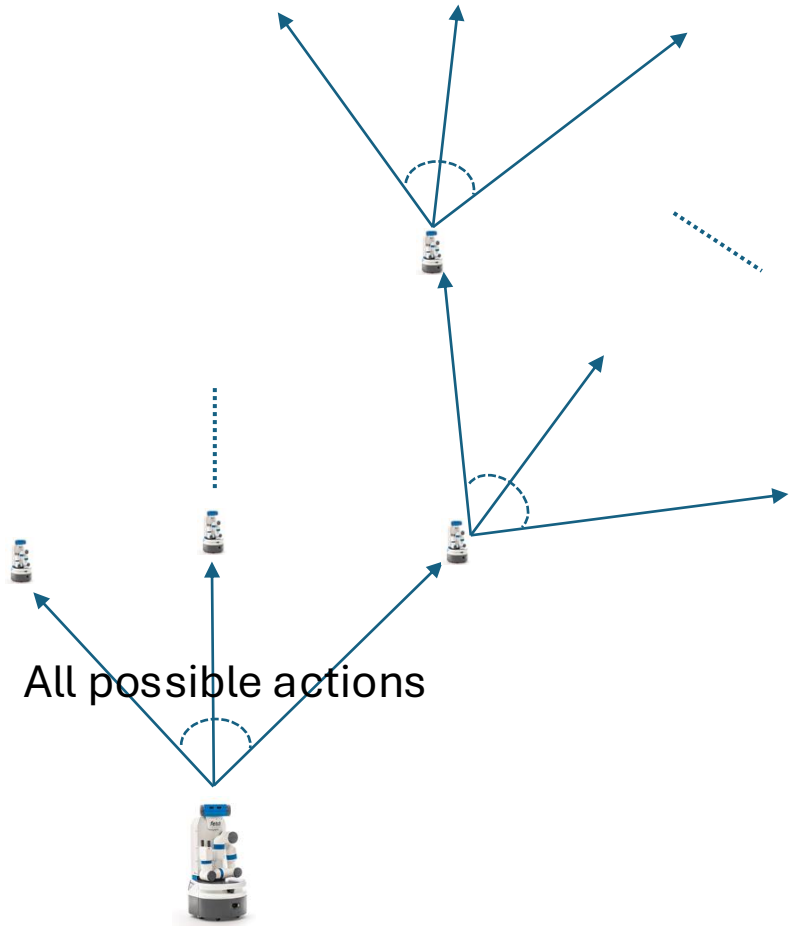
$\ell\big((Arad, Zerind)\big) = \langle drive\_Arad\_Zerind, 75 \rangle$

$\ell\big((Zerind, Arad)\big) = \langle drive\_Zerind\_Arad, 75 \rangle$

Problem with Naïve Formulation of Search Problems

- Given:
  - Graph $\langle V, E, A, \ell \rangle$
    - V set of vertices
    - E set of directed edges
    - A set of actions
    - $\ell: E \rightarrow \langle A, \mathbb{R}^+ \rangle$
  - $v_i \in V$ start vertex
  - G goal vertex set
- Find a path from v_i to G

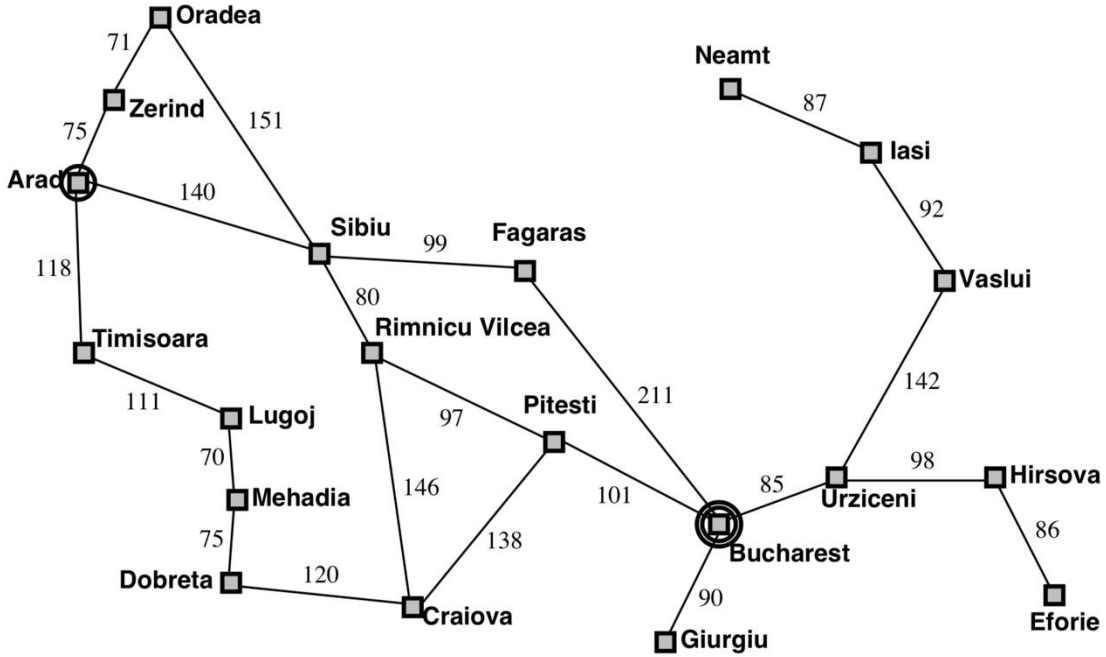
In most cases, the full graph is too large to specify explicitly



Search Problem Defined Over an Implicit Graph

- A search problem is defined by
  - S set of possible states
    - Defined, but not enumerated a priori
  - $s_0 \in S$ start state (~vertex)
  - A set of actions
  - $N: S \rightarrow S$ successor function
    - $T: S \times A \rightarrow S$ deterministic transition function
    - $N(s) = \{s' : \exists a \, T(s, a) = s'\}$

Set of all s' that can be reached in one step from s

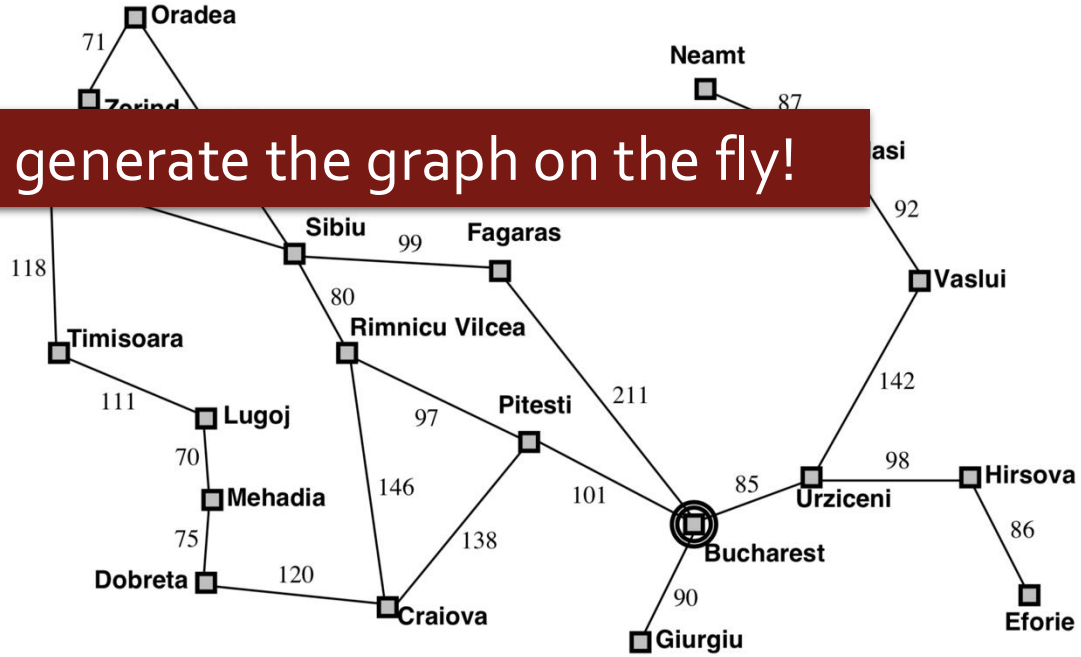


- $N(Arad) = \{Timisoara, Sibiu, Zerind\}$

Search Problem Defined Over an Implicit Graph

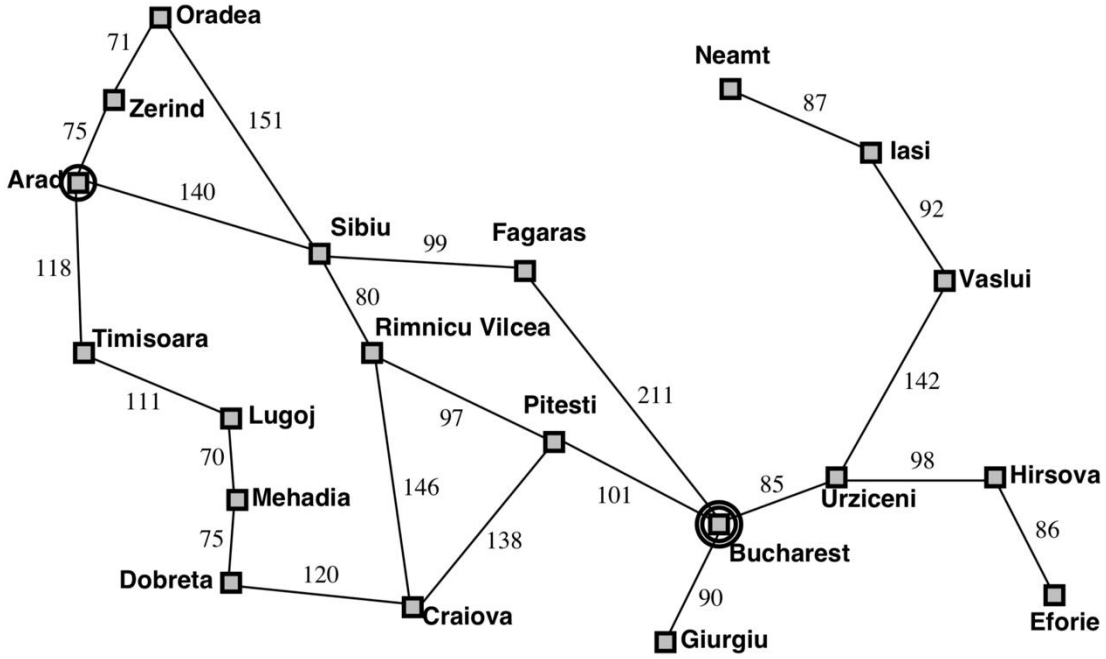
- A search problem is defined by
  - S set of possible states
    - Defined, but not enumerated a priori
  - $s_0 \in S$ start state (~vertex)
  - A set of actions
  - $N: S \to S$ successor function
    - $T: S \times A \to S$ deterministic transition function
    - $N(s) = \{s' : \exists a\, T(s,a) = s'\}$

Allows us to generate the graph on the fly!



Search Problem Defined Over an Implicit Graph

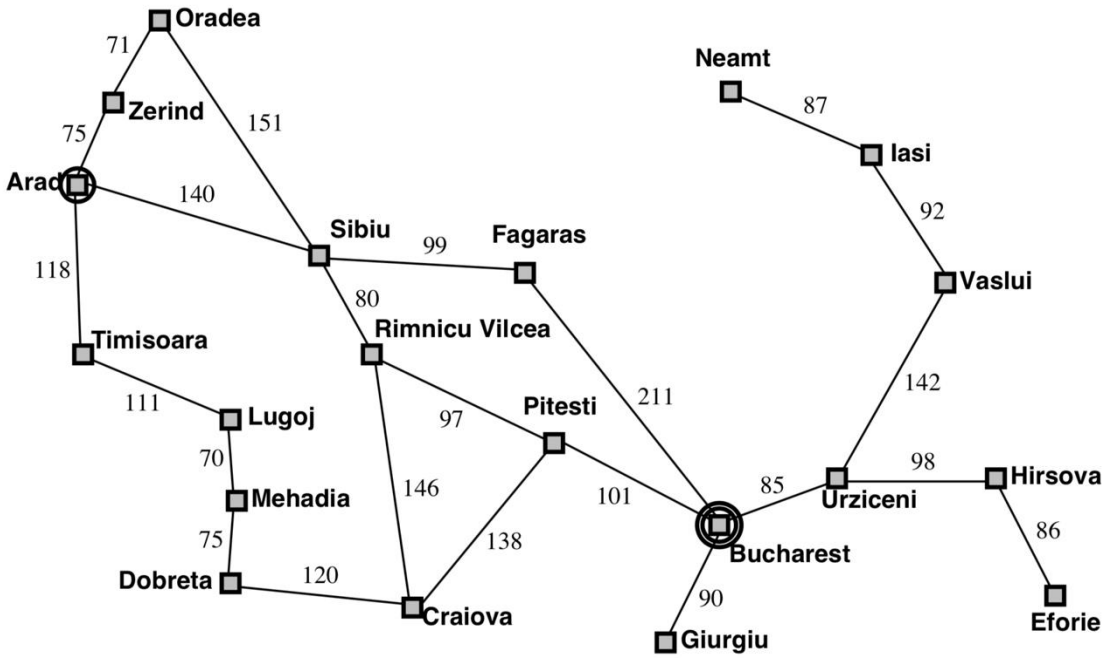
- A search problem is defined by
  - S set of possible states
    - Defined, but not enumerated a priori
  - $s_0 \in S$ start state (~vertex)
  - A set of actions
  - $N: S \rightarrow S$ successor function
    - $T: S \times A \rightarrow S$ deterministic transition function
    - $N(s) = \{s' : \exists a \, T(s, a) = s'\}$
  - $G: S \rightarrow \{True, False\}$ goal test
  - $C: S \times A \times S \rightarrow \mathbb{R}^+$ path cost



A **solution** to a search problem is a sequence of actions leading from s_o to a state satisfying the goal test

Role of Abstraction in Modeling Search Problems

- Search problem formulations are models
  - Most models are abstractions
  - Driving from one city to another involves
    - Traffic, one-way streets, fuel considerations, rest stops, …
- Abstract state = set of real states
- Abstract action = set of procedures

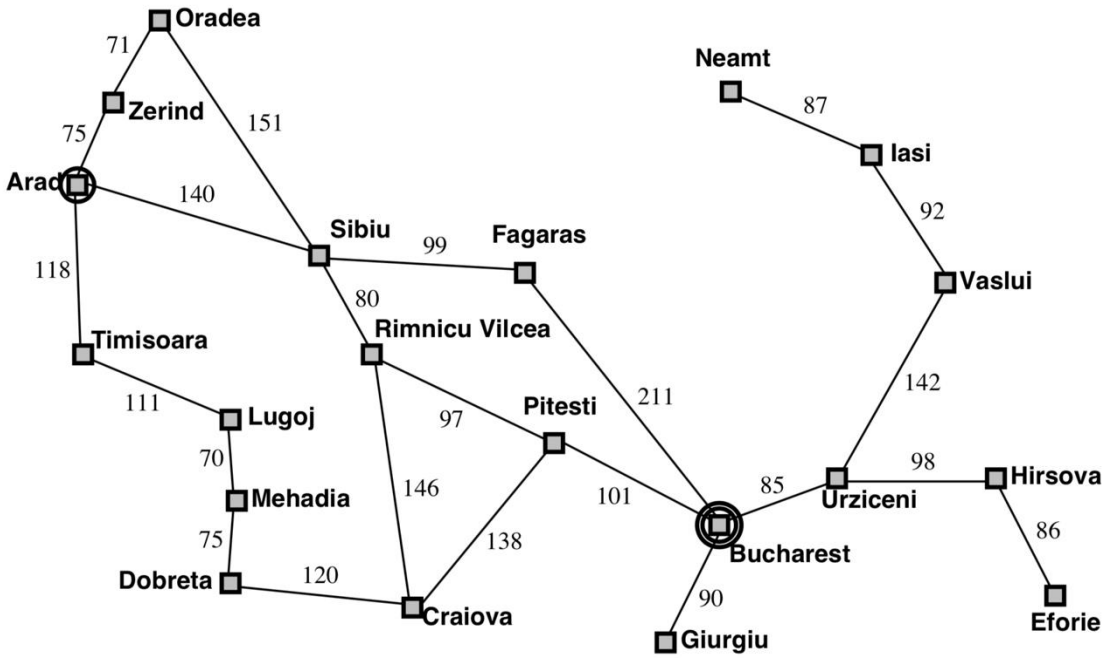


Abstraction is a common and important aspect of modeling

Abstract problem should be easier than the real problem!

Role of Abstraction in Modeling Search Problems

- For the abstract solution to be usable we need:
  - $\forall s, a, s'$ such that $T(s, a) = s'$
    - Every real state in s gets to some real state in s' by following a procedure defined by a
    - "Downward refinability"
  - There should be a way to go from all real locations "in Arad" to some real location "in Zerind"



Abstract problem should be easier than the real problem!